
Subject: CsvComparator application

Posted by [Didier](#) on Fri, 06 Aug 2010 09:10:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi all,

I just made a teaser that uses the TCC lib to compare to cvs files by using a dedicated 'C' comparison code

==> This enables all comparison techniques imaginable and not only the ones available in exel or other.

To launch the app, just compile it and run from console:

CsvComparator ref.csv csvComparaisionCode.h result.csv

The files ref.csv and result.csv are compared using 'csvComparaisionCode.h'.

In the display you can see the columns beeing compared are in black while others , which are not being compared are in light gray.

When a cell is in default ==> the difference is displayed in red

How the comparison file is build:

```
// header data: must always be there
```

```
typedef enum {  
    CDTE_NOT_PROCESSED= 0,  
    CDTE_UINT      = 1,  
    CDTE_INT       = 2,  
    CDTE_FLOAT     = 3,  
    CDTE_STRING    = 4,  
    CDTE_BOOL      = 5  
} ColumnDataTypeEnum;
```

```
typedef unsigned long long CDT_UINT;
```

```
typedef signed long long  CDT_INT;
```

```
typedef double           CDT_FLOAT;
```

```
typedef const char*      CDT_STRING;
```

```
typedef bool             CDT_BOOL;
```

```
// enables to set the line and column separators used in the csv files
```

```
// if this function is not present, the default values are 'space' and 'tab'
```

```
void getParsingSeparators(char* lineSeparator, char* columnSeparator)
```

```
{  
    *lineSeparator='\n';  
    *columnSeparator=' ';  
}
```

```
// this function enables to set the columns beeing processed (== compared)
// by setting their data type
```

```
ColumnDataTypeEnum getDataType(int colNbr)
{
    ColumnDataTypeEnum res = CDTE_NOT_PROCESSED;
    switch(colNbr)
    {
        case 0: res = CDTE_INT; break;
        case 1: res = CDTE_STRING; break;
        case 2: res = CDTE_FLOAT; break;
        case 3: res = CDTE_FLOAT; break;
        case 7: res = CDTE_STRING; break;
    }
    return res;
}
```

```
// For each column that needs to be processed, there is a corresponding testcol_xxx function
// * The 'xxx' value is the same as in the 'switch case' of the 'getDataType' function
// * The INPUT DATA TYPES of the comparison functions corresponds to the type defined by 'getDataType'
```

```
// * The other parameters 'reslutStrings' and 'refStrings' allow a comparison function to access data on other
```

```
// columns in text form
```

```
bool testCol_0(int curRow, CDT_INT* col, CDT_INT* refCol, const char*** reslutStrings, const char*** refStrings)
```

```
{
    return (col[curRow] == refCol[curRow]);
}
```

```
bool testCol_1(int curRow, CDT_STRING* col, CDT_STRING* refCol, const char*** reslutStrings, const char*** refStrings)
```

```
{
    return ( strcmp(col[curRow], refCol[curRow]) == 0 );
}
```

```
bool testCol_2(int curRow, CDT_FLOAT* col, CDT_FLOAT* refCol, const char*** reslutStrings, const char*** refStrings)
```

```
{
    return ((col[curRow] == refCol[curRow]) || (col[curRow] == -refCol[curRow]));
}
```

```
bool testCol_3(int curRow, CDT_FLOAT* col, CDT_FLOAT* refCol, const char*** reslutStrings, const char*** refStrings)
```

```
{
```

```
return (col[curRow] == refCol[curRow]);  
}
```

```
bool testCol_7(int curRow, CDT_STRING* col, CDT_STRING* refCol, const char*** reslutStrings,  
const char*** refStrings)  
{  
    return ( strcmp(col[curRow], refCol[curRow]) == 0 );  
}
```

This is just a programm I made for fun(and therefore is far from beeing perfect), but the use of Tcc is quite interesting.

Maybe some of you might find this useful.

NB: it compiles under windows and linux, but works only on linux du to include/Tcc problems. I didn't take time to look into this problem.

File Attachments

1) [CsvComparator.tar.gz](#), downloaded 446 times

Subject: Re: CsvComparator application

Posted by [Didier](#) on Fri, 06 Aug 2010 09:13:25 GMT

[View Forum Message](#) <> [Reply to Message](#)

A sample screenshot:

File Attachments

1) [CsvComparator.png](#), downloaded 1013 times

Subject: Re: CsvComparator application

Posted by [koldo](#) on Mon, 06 Sep 2010 10:32:34 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Didier

Sorry for the late answer. The program works perfectly in Windows too with MinGW without (almost) any change.

Now in main.cpp, method doCompare(), there is:

```
tcc.AddIncludePath("/usr/include");  
tcc.AddLibraryPath("/usr/lib");just put the adequate paths for Windows like "Tcc\\lib\\include" and  
"Tcc\\lib\\lib".
```

It is not possible to compile MSC as it does not let declarations with dynamic allocation:

```
T columnData[nbRows];
```

Some comments:

- In main.cpp, line 254, it seems to have a call to `getData` without arguments
- In `GUI_APP_MAIN`, when run without arguments, its called 12 times `TRACE_INFO()` that calls `PromptOK()`.

File Attachments

1) [dib.PNG](#), downloaded 934 times

Subject: Re: CsvComparator application
Posted by [Didier](#) on Tue, 07 Sep 2010 10:10:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Koldo,

thanks for looking at this small app.

Maybe it could serve as an extended TCC example ?

Subject: Re: CsvComparator application
Posted by [koldo](#) on Tue, 07 Sep 2010 11:25:10 GMT
[View Forum Message](#) <> [Reply to Message](#)

Didier wrote on Tue, 07 September 2010 12:10Hi Koldo,

thanks for looking at this small app.

Maybe it could serve as an extended TCC example ?

Hello Didier

Maybe. Perhaps the interface would be easier as now the files to be compared have to be introduced in a command line.

For me it would be easier to have the files to be compared and the source code in gui.
